Shortest Paths

Dijkstra’s Algorithm

- Finds the shortest (lowest-cost) path in a graph from start node to all other nodes

- Is the fastest, single start, shortest path algorithm for directed and undirected graphs with unbounded, non-negative edge weights

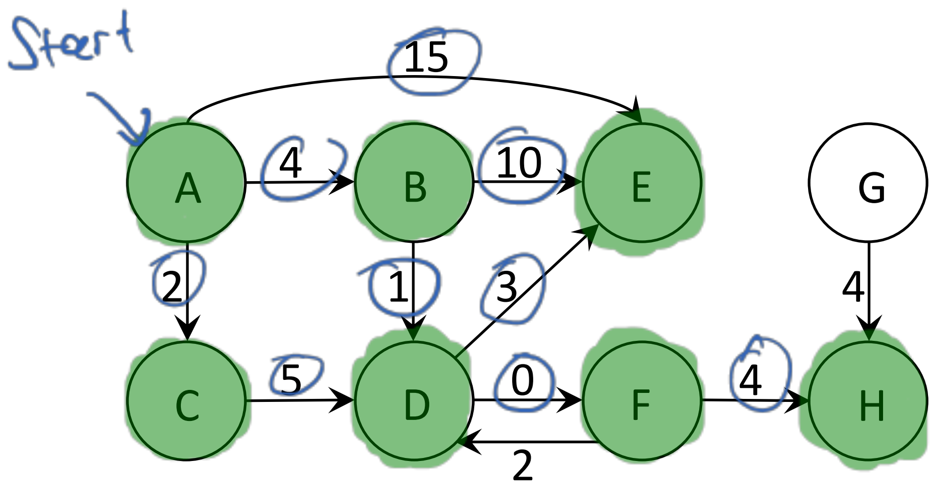
Dijkstra Pseudocode

path: [destination, predecessor, cost]

Starting node

```
dijkstra(v):
    pq = new PriorityQueue()
    pq.insert(  [ dest:v, pred:null, cost:0 ] )
    while ( !pq.isEmpty() ):
        [ dest, pred, cost ] = pq.removeMin()
        if dest is unvisited:
            mark dest as visited, store pred and cost for dest
            for each edge with weight w to unvisited neighbor u of dest:
                pq.insert( [ u, dest, cost + w ] )
```

Example



vertex	visited	pred	cost
A	~~F~~ T	null	0
B	~~F~~ T	A	4
C	~~F~~ T	A	2
D	~~F~~ T	B	5
E	~~F~~ T	D	8
F	~~F~~ T	D	5
G	F	—	—
H	~~F~~ T	F	9

priority queue: ~~[A, null, 0]~~, ~~[B, A, 4]~~, ~~[C, A, 2]~~, ~~[E, A, 15]~~, ~~[D, C, 7]~~, ~~[D, B, 5]~~, ~~[E, B, 14]~~, ~~[E, D, 8]~~, ~~[F, D, 5]~~, ~~[H, F, 9]~~

Reconstructing Paths

- Path A to E:
  path cost : 8
  path: ABDE
- Path A to F:
  path cost: 5
  path: ABDF
- Path A to G:
  path cost : ______
  path: ______

Complexity

V: # nodes

E: # edges

Maximum number of paths in priority queue: E

Adding / removing a path from priority queue: $\log E$

$\Rightarrow O(E \cdot \cancel{2} \cdot \log E)$

$O(E \cdot \log V^2)$

$\rightarrow O(E \cdot \cancel{2} \cdot \log V)$